

Network of Experts for Large-Scale Image Categorization

Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani

Department of Computer Science,
Dartmouth College
{karim, haris}@cs.dartmouth.edu, LT@dartmouth.edu

Abstract. We present a tree-structured network architecture for large-scale image classification. The trunk of the network contains convolutional layers optimized over all classes. At a given depth, the trunk splits into separate branches, each dedicated to discriminate a different subset of classes. Each branch acts as an expert classifying a set of categories that are difficult to tell apart, while the trunk provides common knowledge to all experts in the form of shared features. The training of our “network of experts” is completely end-to-end: the partition of categories into disjoint subsets is learned simultaneously with the parameters of the network trunk and the experts are trained jointly by minimizing a single learning objective over all classes. The proposed structure can be built from any existing convolutional neural network (CNN). We demonstrate its generality by adapting 4 popular CNNs for image categorization into the form of networks of experts. Our experiments on CIFAR100 and ImageNet show that in every case our method yields a substantial improvement in accuracy over the base CNN, and gives the best result achieved so far on CIFAR100. Finally, the improvement in accuracy comes at little additional cost: compared to the base network, the training time is only moderately increased and the number of parameters is comparable or in some cases even lower.

Keywords: Deep learning, convolutional networks, image classification.

1 Introduction

Our visual world encompasses tens of thousands of different categories. While a layperson can recognize effectively most of these visual classes [4], discrimination of categories in specific domains requires expert knowledge that can be acquired only through dedicated training. Examples include learning to identify mushrooms, authenticate art, diagnosing diseases from medical images. In a sense, the visual system of a layperson is a very good generalist that can accurately discriminate coarse categories but lacks the specialist eye to differentiate fine categories that look alike. Becoming an expert in any of the aforementioned domains involves time-consuming practical training aimed at specializing our visual system to recognize the subtle features that differentiate the given classes.

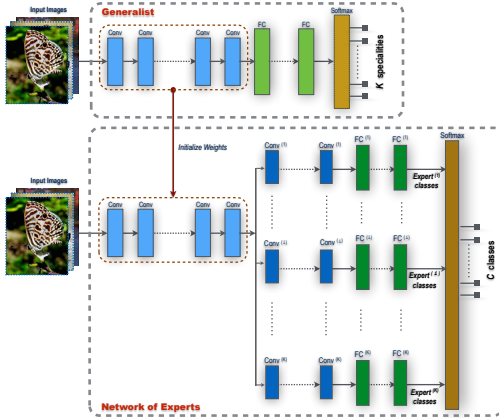


Fig. 1. Our Network of Experts (NOFE). **Top:** Training of the generalist. The generalist is a traditional CNN but it is optimized to partition the original set of C classes into $K \ll C$ disjoint subsets, called specialties. Our method performs *joint* learning of the K specialties and the generalist CNN that is optimized to recognize these specialties. **Bottom:** The complete NOFE with K expert branches. The convolutional layers of the generalist are used as initialization for the trunk, which ties into K separate branches, each responsible to discriminate the classes within a specialty. The complete model is trained end-to-end via backpropagation with respect to the original C classes.

Inspired by this analogy, we propose a novel scheme that decomposes large-scale image categorization into two separate tasks: 1) the learning of a generalist optimized to discriminate coarse groupings of classes, i.e., disjoint subsets of categories which we refer to as “specialties” and 2) the training of experts that learn specialized features aimed at accurate recognition of classes within each specialty. Rather than relying on a hand-designed partition of the set of classes, we propose to *learn* the specialties for a substantial improvement in accuracy (see Fig. 2). Our scheme simultaneously learns the specialties and the generalist that is optimized to recognize these specialties. We frame this as a joint minimization of a loss function $E(\theta^G, \ell)$ over the parameters θ^G of the generalist and a labeling function ℓ that maps each original category to a specialty. In a second training stage, for each specialty, an expert is trained to classify the categories within that specialty.

Although our learning scheme involves two distinct training stages – the first aimed at learning the generalist and the specialties, the second focused on training the experts – the final product is a unified model performing multi-class classification over the original classes, which we call “Network of Experts” (NOFE). The training procedure is illustrated in Fig 1. The generalist is implemented in the form of a convolutional neural network (CNN) with a final softmax layer over K specialties, where $K \ll C$, with C denoting the original number of categories (Figure 1(top)). After this first training stage, the fully connected layers are discarded and K distinct branches are attached to the last

convolutional layer of the generalist, i.e., one branch per specialty. Each branch is associated to a specialty and is devoted to recognize the classes within the specialty. This gives rise to the NOFE architecture, a unified tree-structured network (Figure 1(bottom)). Finally, all layers of the resulting model are fine-tuned with respect to the original C categories by means of a global softmax layer that calibrates the outputs of the individual experts over the C categories.

Thus, the learning of our generalist serves two fundamental purposes:

- 1) First, using a *divide and conquer* strategy it decomposes the original multi-class classification problem over C labels into K subproblems, one for each specialty. The specialties are defined so that the act of classifying an image into its correct specialty is as accurate as possible. At the same time this implies that confusable classes are pushed into the same specialty, thus handing off the most challenging class-discrimination cases to the individual experts. However, because each expert is responsible for classification only over a subset of classes that are highly similar to each other, it can learn highly specialized and effective features for the subproblem, analogously to a human expert identifying mushrooms by leveraging features that are highly domain-specific (cap shape, stem type, spore color, flesh texture, etc.).
- 2) Second, the convolutional layers learned by the generalist provide an initial knowledge-base for all experts in the form of shared features. In our experiments we demonstrate that fine-tuning the trunk from this initial configuration results in a significant improvement over learning the network of experts from scratch or even learning from a set of convolutional layers optimized over the entire set of C labels. Thus, the subproblem decomposition does not merely simplify the original hard classification problem but it also produces a set of pretrained features that lead to better finetuning results.

We note that we test our approach on image categorization problems involving a large number of classes, such as ImageNet classification, where the classifier must have the ability to recognize coarse categories (“vehicle”) but must also distinguish highly confusable specialty classes (e.g., “English pointer” from “Irish setter”). These scenarios match well the structure of our model, which combines a generalist with a collection of experts. We do not assess our approach on a fine-grained categorization benchmark as this typically involves classification focused only on one domain (say, bird species) and thus does not require the generalist and multiple specialists learned by our model.

2 Related Work

Our work falls in the category of CNN models for image classification. This genre has witnessed dramatic growth since the introduction of the “AlexNet” network [23]. In the last few years further recognition improvements have been achieved thanks to advances in CNN components [11,28,15,13] and training strategies [25,9,24,18,16]. Our approach instead achieves gains in recognition accuracy by means of an architectural alteration that involves adapting exist-

ing CNNs into a tree-structure. Thus, our work relates to prior studies of how changes in the network structure affect performance [21,14].

Our adaptation of base CNN models into networks of experts hinges on a method that groups the original classes into specialties, representing subsets of confusable categories. Thus, our approach relates closely to methods that learn hierarchies of categories. This problem has received ample study, particularly for the purpose of speeding up multi-class classification in problems involving large number of categories [12,29,2,10,8,27]. Hierarchies of classes have also been used to infer class abstraction [19], to trade off concept specificity versus accuracy [8,31], to allow rare objects to borrow statistical strength from related but more frequent objects [32,36], and also for unsupervised discovery of objects [34].

Our proposed work is most closely related to methods that learn groupings of categories in order to train expert CNNs that specialize in different visual domains. Hinton et al. [17] introduced an ensemble network composed of one or more full models and many specialist models which learn to distinguish fine-grained classes that the full models confuse. Similarly, Warde-Farley et al. [37] augment a very large CNN trained over all categories via auxilliary hidden layer pathways that connect to specialists trained on subsets of classes. Yan et al. [38] presented a hierarchical deep CNN (HD-CNN) that consists of a coarse component trained over all classes as well as a set of fine components trained over subsets of classes. The coarse and the fine components share low-level features and their predictions are late-fused via weighted probabilistic averaging. While our approach is similar in spirit to these three expert-systems, it differs substantially in terms of architecture and it addresses some of their shortcomings:

1. In [17,37,38] the experts are learned only after having trained a large-capacity CNN over the original multi-class classification problem. The training of our approach does not require the expensive training of the base CNN model over all C classes. Instead it directly learns a generalist that discriminates a much smaller number of specialties ($K \ll C$). The experts are then trained as categorizers within each specialty. By using this simple *divide and conquer* the training cost remains manageable and the overall number of parameters can be even lower than that of the base model (see Table 5).
2. The architectures in [17,38] route the input image to only a subset of experts, those deemed more competent in its categorization. A routing mistake cannot be corrected. To minimize this error, redundancy between experts must be built by using overlapping specialties (i.e., specialties sharing classes) thus increasing the number of classes that each expert must recognize. Instead, in our approach the specialties are disjoint and thus more specific. Yet, our method does not suffer from routing errors as all experts are invoked in parallel for each input image.
3. Although our training procedure involves two distinct stages, the final phase performs fine-tuning of the complete network of experts using a single objective over the original C categories. While fine-tuning is in principle possible for both [17] and [37], in practice this was not done because of the large computational cost of training and the large number of parameters.

3 Technical approach

In this section we present the details of our technical approach. We begin by introducing the notation and the training setup.

Let $\mathcal{D} = \{(x^1, y^1), \dots, (x^N, y^N)\}$ be a training set of N class-labeled images where $x^i \in \mathbb{R}^{r \times c \times 3}$ represents the i -th image (consisting of r rows, c columns and 3 color channels) and $y^i \in \mathcal{Y} \equiv \{1, 2, \dots, C\}$ denotes its associated class label (C denotes the total number of classes).

Furthermore, we assume we are given a CNN architecture $b_{\theta^B} : \mathbb{R}^{r \times c \times 3} \rightarrow Y$ parameterized by weights θ^B that can be optimized to categorize images into classes Y . We refer to this CNN model as the *base* architecture, since we will use this architecture to build our network of experts resulting in a classifier $e_{\theta^E} : \mathbb{R}^{r \times c \times 3} \rightarrow Y$. In our empirical evaluation we will experiment with different choices of base classifiers [23, 33, 14]. Here we abstract away the specificity of individual base classifiers by assuming that b_{θ^B} consists of a CNN with a certain number of convolutional layers followed by one or more fully connected layers and a final softmax layer that defines a posterior distribution over classes in \mathcal{Y} . Finally, we assume that the parameters of the base classifier can be learned by optimizing an objective function of the form:

$$E_b(\theta; \mathcal{D}) = R(\theta) + \frac{1}{N} \sum_{i=1}^N L(\theta; x^i, y^i) \quad (1)$$

where R is a regularization term aimed at preventing overfitting (e.g., weight decay) and L is a loss function penalizing misclassification (e.g., the cross entropy loss). As \mathcal{D} is typically large, it is common to optimize this objective using back-propagation over mini-batches, i.e., by considering at each iteration a random subset of examples $\mathcal{S} \subset \mathcal{D}$ and then minimizing $E_b(\theta; \mathcal{S})$.

In the following subsections we describe how to adapt the architecture of the base classifier b_{θ^B} and the objective function E_b in order to learn a network of experts. Note that our approach does not require *learning* (i.e., optimizing) the parameters of the base classifier (i.e., optimizing parameters θ^B). Instead it simply needs a base CNN architecture (with uninstantiated weights) and a learning objective. We decompose the training into two stages: the learning of the generalist (described in subsection 3.1) and the subsequent training of the complete network of experts (presented in subsection 3.2), which uses the generalist as initialization for the trunk and the definition of the specialties.

3.1 Learning the Generalist

The goal of this first stage of training is to learn groupings of classes, which we call specialties. Intuitively, we want each specialty to represent a subset of classes that are highly confusable (such as different mushrooms) and that, as such, require the specialized analysis of an expert. Formally, the specialties represent a partition of the set Y . In other words, the specialties are $K \ll C$ disjoint subsets of classes whose union gives \mathcal{Y} and where K represents a hyperparameter defining

the number of experts and thus the complexity of the system. We can cast the definition of the specialties as the problem of learning a label mapping $\ell : \mathcal{Y} \rightarrow \mathcal{Z}$, where $\mathcal{Z} = \{1, 2, \dots, K\}$ is the set of specialty labels. Conceptually we want to define ℓ such that we can train a generalist $g_{\theta^G} : \mathbb{R}^{r \times c \times 3} \rightarrow \mathcal{Z}$ that correctly classifies image x^i into its associated specialty, i.e., such that $g(x^i; \theta^G) = \ell(y^i)$. We formulate this task as a joint optimization over the parameters θ^G of the generalist and the mapping ℓ so as to produce the best possible recognition accuracy over the specialty labels by considering the objective

$$E_g(\theta^G, \ell; \mathcal{D}) = R(\theta) + \frac{1}{N} \sum_{i=1}^N L(\theta; x^i, \ell(y^i)). \quad (2)$$

Note that this is the same objective as in Eq. 1, except that the labels of the examples are now defined in terms of the mapping ℓ , which is itself unknown. Thus we can now view this learning objective as a function over unknown parameters θ^G, ℓ . The architecture of the generalist is the same as that of the base model except for the use of a softmax over \mathcal{Z} instead of \mathcal{Y} and for the dimensionality of the last fully connected layer, which also needs to change in order to match the number of specialties, K .

We optimize this objective via a simple alternation scheme that iterates between the following two steps:

1. Optimizing parameters θ^G while keeping specialty labels ℓ fixed.
2. Updating specialty labels ℓ given the current estimate of weights θ^G .

First, we initialize the mapping ℓ by randomly partitioning Y into K subsets, each containing C/K classes (in all our experiments we use values of K that are factors of C so that we can produce a set of K perfectly-balanced specialties). Given this initial set of specialty labels, the first step of the alternation scheme is implemented by running several iterations of stochastic gradient descent.

The second step of our alternation requires optimizing with respect to ℓ given the current estimate of parameters θ^G . For this purpose, we evaluate the generalist defined by the current parameters θ^G over a random subset $\mathcal{S} \subset \mathcal{D}$ of the training data. For this set we build the confusion matrix $M \in \mathbb{R}^{C \times K}$, where M_{ij} is the fraction of examples of class label i that are classified into specialty j by the *current* generalist. Then, a greedy strategy would be to set $\ell(i) = \arg \max_{j \in \{1, \dots, K\}} M_{ij}$ for each class $i \in \{1, \dots, C\}$ so that each class is assigned to the specialty that recognizes the maximum number of images of that class. Another solution is to perform spectral clustering over the confusion matrix, as done in [38, 17, 3]. However, we found that both of these solutions yield highly imbalanced specialty clusters, where a few specialties absorb nearly all classes, making the problem of classification within these large specialties almost as hard as the original, as confirmed in our experiments. To address this problem we tried two different schemes that either constrain or softly encourage the specialties to have an equal number of classes, as discussed next.

- The first scheme, which we refer to as **fully-balanced** forces the specialties to have equal size. Initially the specialties are set to be empty and they are

then grown by considering the classes in \mathcal{Y} one at a time, in random order. For each class $i \in \mathcal{Y}$, we assign the specialty j that has the highest value $M_{i,j}$ among the specialties that have not yet reached maximum size C/K . The randomization in the visiting order of classes guarantees that, over multiple label updates, no class is favored over the others.

- Unlike the previous scheme which produces perfectly balanced specialties, **elasso** is a method that allows us to encourage *softly* the constraint over the size of specialties. This may be desirable in scenarios where certain specialties should be allowed to include more classes than others. The procedure is adapted from the algorithm of Chang et al. [5]. To define the specialties for this method we use a clustering indicator matrix $F \in \{0, 1\}^{C \times K}$ where each row of F has one entry only set to 1, denoting the specialty assigned to the class. Let us indicate with $Ind(C, K)$ the set of all clustering indicator matrices of size $C \times K$ that satisfy this constraint. In order to create specialties that are simultaneously easy to classify and balanced in size, we compute F by minimizing the objective

$$\min_{F \in Ind(C, K)} \lambda \|F\|_e - \|M \odot F\|_{1,1} \quad (3)$$

where $\|F\|_e = \sqrt{\sum_j (\sum_i F_{ij})^2}$ is the so-called exclusive lasso norm [5,39], \odot denotes the element-wise product between matrices, $\|A\|_{1,1} = \sum_i \sum_j |A_{ij}|$ is the $L_{1,1}$ -norm, and λ is a hyperparameter trading off the importance between having balanced specialties and good categorization accuracy over them. Note that the first term captures the balance degree of the specialties: for each j , it computes the squared-number of classes assigned to specialty j and then sums these squared-numbers over all specialties. Thus, $\|F\|_e$ uses an L_1 -norm to compute the number of classes assigned to each specialty, and then an L_2 -norm to calculate the average size of the specialty. The L_2 -norm strongly favors label assignments that generate specialties of roughly similar size. The second term, $\|M \odot F\|_{1,1} = \sum_j \sum_i M_{ij} F_{ij}$, calculates the *accuracy* of specialty classification. As we want to make specialty classification accuracy as high as possible, we subtract this term from the exclusive lasso norm to define a *minimization* objective. As in [5], we update one row of F at a time. Starting from an initial F corresponding to the current label mapping ℓ , we loop over the rows of F in random order and for each row we find the element being 1 that yields the minimum of Eq. 3. This procedure is repeated until convergence (see [5] for a proof of guaranteed convergence).

3.2 Training the Network of Experts

Given the generalist θ^G and the class-to-specialty mapping ℓ produced by the first stage of training, we perform joint learning of the K experts in order to obtain a global multi-class classification model over the original categories in the label set \mathcal{Y} . As illustrated in Fig.1, this is achieved by defining a tree-structured network consisting of a single trunk feeding K branches, one branch for each specialty. The trunk is initialized with the convolutional layers of the generalist, as

they have been optimized to yield accurate specialty classification. Each branch contains one or more convolutional layers followed by a number of fully-connected layers (in our experiments we set the expert to have as many fully connected layers as the base model). However, each branch is responsible for discriminating only the classes associated to its specialty. Thus, the number of output units of the last fully-connected layer is equal to the number of classes in the specialty (this is exactly equal to C/K for **fully-balanced**, while it varies for individual specialties in **elasso**). The final fully-connected layer of each branch is fed into a *global* softmax layer defined over the entire set of C labels in the set \mathcal{Y} . This softmax layer does not contain weights. Its purpose is merely to normalize the outputs of the K experts to define a proper class posterior distribution over \mathcal{Y} .

The parameters of the resulting architecture are optimized via backpropagation with respect to the training set \mathcal{D} and labels in \mathcal{Y} using the regularization term R and loss function L of the base model. This implies that for each example x^i both forward and backward propagation will run in all K branches, irrespective of the ground truth specialty $\ell(y^i)$ of the example. The backward pass from the ground-truth branch will aim at increasing the output value of the correct class y^i , while the backward pass from the other $K - 1$ branches will update the weights to lower the probabilities of their classes. Because of this joint training the outputs of the experts will be automatically calibrated to define a proper probability distribution over \mathcal{Y} . While the weights in the branches are randomly initialized, the convolutional layers in the trunk are initially set to the parameters computed by the generalist. Thus, this final learning stage can be viewed as performing fine-tuning of the generalist parameters for the classes in \mathcal{Y} using the network of experts.

Given the learned NOFE, inference is done via forward propagation through the trunk and all K branches so as to produce a full distribution over \mathcal{Y} .

4 Experiments

We performed experiments on two different datasets: CIFAR100 [22], which is a medium size dataset, and the large-scale ImageNet benchmark [7].

4.1 Model Analysis on CIFAR100

The advantage of CIFAR100 is that its medium size allows us to carry out a comprehensive study of many different design choices and architectures, which would not be feasible to perform on the large-scale ImageNet benchmark. CIFAR100 consists of color images of size 32x32 categorized into 100 classes. The training set contains 50,000 examples (500 per class) while the test set consists of 10,000 images (100 per class).

Our first set of studies are performed using as base model $b_{\theta B}$ a CNN inspired by the popular AlexNet model [23]. It differs from AlexNet in that it uses 3 convolutional layers (instead of 5) and 1 fully connected layer (instead of 3) to work

on the smaller-scale CIFAR100 dataset. We call this smaller network AlexNet-C100. The full specifications of the architecture are given in the supplementary material, including details about the learning policy.

Our generalist is identical to this architecture with the only difference being that we set the number of units in the FC and SM layers to K , the number of specialties. The training is done from scratch. The learning alternates between updating network parameters θ^G and specialty labels ℓ . The specialty labels are updated every 1 epoch of backpropagation over θ^G . We use a random subset \mathcal{S} of 10,000 images to build the confusion matrix. In the supplementary material we show some of the specialties learned by our generalist. Most specialties define intuitive clusters of classes, such as categories that are semantically or visually similar to each other (e.g., dolphin, seal, shark, turtle, whale).

Once the generalist is learned, we remove its FC layer and connect K branches, each consisting of: [CONV:1×64×5],[FC:c] where [CONV:1×64×5] denotes 1 convolutional layer containing 64 filters of size 5×5 , [FC:c] is a fully connected layer with c output units corresponding to the classes in the specialty (note that c may vary from specialty to specialty). We link the K FC layers of the branches to a *global* softmax over all C classes (without parameters). The weights of each branch are randomly initialized. The full NOFE is trained via backpropagation using the same learning rate policy as for the base model.

Number of Experts and Specialty Balance. The degree of specialization of the experts in our model is controlled by parameter K . Here we study how the value of this hyperparameter affects the final accuracy of the network. Furthermore, we also assess the importance of balancing the size of the specialties in connection with the value of K , since these two factors are interdependent. The method **fully-balanced** (introduced in section 3.1) constrains all specialties to have equal size (C/K), while **elasso** encourages softly the constraint over the size of specialties. The behavior of **elasso** is defined by hyperparameter λ which trades off the importance between having balanced specialties and good categorization accuracy over them.

Table 1 summarizes the recognition performance of our NOFE for different values of K and the two ways of balancing specialty sizes. For **elasso** we report accuracy using $\lambda = 1000$, which was the best value according to our evaluation. We can immediately see that the two balancing methods produce similar recognition performance, with **fully-balanced** being slightly better than **elasso**. Perhaps surprisingly, **elasso**, which gives the freedom of learning specialties of unequal size, is overall slightly worse than **fully-balanced**. From this table we can also evince that our network of experts is fairly robust to the choice of K . As K is increased the accuracy of each balancing method produces an approximate “inverted U” curve with the lowest performance at the two ends ($K = 2$ and $K = 50$) and the best accuracy for $K = 5$ or $K = 10$. Finally, note that all instantiations of our network of experts in this table achieve higher accuracy than the “flat” base model, with the exception of the models using $K = 2$ experts which provide performance comparable to the baseline. Our best model ($K = 10$

Table 1. Top-1 accuracy (%) on CIFAR100 for the base model (AlexNet-C100) and Network of Experts (NoFE) using varying number of experts (K) and two different specialty balancing methods. The best NoFE outperforms the base model by 2.2% and all NoFE using $K > 2$ experts yield better accuracy than the flat architecture.

Model	Balancing Method	K=2	K=5	K=10	K=20	K=50
NoFE	fully-balanced elasso $\lambda=1000$	53.3 53.93	55.0 53.6	56.2 55.59	55.7 55.3	55.33 55.3
Base: AlexNet-C100	n/a	54.0				

Table 2. CIFAR100 top-1 accuracy of NoFE models trained on different definitions of specialties: ours, spectral clusters of classes and random specialties of equal size.

Specialty Method	Accuracy %
Our Method (joint training)	56.2
Spectral Clustering	53.2
Random Balanced Specialties	53.7

Table 3. CIFAR100 top-1 accuracy (%) for 5 different CNN base architectures and corresponding NoFE models. In each of the five cases our NoFE yields improved performance.

Architecture	Base Model	NoFE
AlexNet-C100	54.04	56.24
AlexNet-Quick-C100	37.94	45.58
VGG11-C100	68.48	69.27
NIN-C100	64.73	67.96
ResNet56-C100	73.52	76.24

using **fully-balanced**) yields a substantial improvement over the base model (56.2% versus 54.0%) corresponding to a relative gain in accuracy of about 4%.

Based on the results of Table 1, all our subsequent studies are based on a NoFE architecture using $K = 10$ experts and **fully-balanced** for balancing.

Defining Specialties with Other Schemes. Here we study how the definition of specialties affects the final performance of the NoFE. Prior work [17,37,38] has proposed to learn groupings of classes by first training a CNN over all C classes and then performing spectral clustering [30] of the confusion matrix. We tried this procedure by building the confusion matrix using the predictions of the base model (AlexNet-C100) over the entire CIFAR100 training set. We then partitioned the $C = 100$ classes into $K = 10$ clusters using spectral clustering. We learned a generalist optimized to categorize these K clusters (without any update of the specialty labels) and then a complete NoFE. The performance of the resulting NoFE is illustrated in the second row of Table 2. The accuracy is considerably lower than when learning specialties with our approach (first row). The third row shows accuracy achieved when training the generalist and subsequently the full NoFE on a random partitioning of the classes into $K = 10$ clusters of equal size. The performance is again inferior to that achieved with our approach. Yet, surprisingly it is better than the accuracy produced by spectral clustering. We believe that this happens because spectral clustering yields highly imbalanced clusters that lead to poor performance of the NoFE.

Varying the Base CNN. The experiments above were based on AlexNet-C100 as the base model. Here we study the generality of our approach by considering 4 other base CNNs (see supplementary material for architecture details):

- 1) *AlexNet-Quick-C100*. This base model is a slightly modified version of AlexNet-C100 where we added an extra FC layer (with 64 output units) before the existing FC layer and removed local response normalization. This leads to much faster convergence but lower accuracy compared to AlexNet-C100. After training the generalist, we discard the two FC layers and attach K branches, each with the following architecture: [CONV:1×64×5], [FC:64],[FC:10].
- 2) *VGG11-C100*. This model is inspired by the VGG11 architecture described in [33] but it is a reduced version to work on the smaller-scale CIFAR100. We take this model from [20]. In the expert branch we use 2 convolutional layers and 3 FC layers to match the number of FC layers in the base model but we scale down the number of units to account for the multiple branches. The branch architecture is: [CONV:2×256×3],[FC:512],[FC:512],[FC:10].
- 3) *NIN-C100*. This is a “Network-In-Network” architecture [26] that was used in [38] as base model to train the hierarchical HD-CNN network on CIFAR100.
- 4) *ResNet56-C100*. This is a residual learning network [14] of depth 56 using residual blocks of 2 convolutional layers. We modeled it after the ResNet-56 that the authors trained on CIFAR10. To account for the 10X number of classes in CIFAR100 we quadruple the number of filters in each convolutional layer. To maintain the architecture homogenous, each expert branch consists of a residual block (rather than a CONV layer) followed by average pooling and an FC layer.

These 4 NOFE models were trained using $K = 10$ and **fully-balanced**. The complete results for these 4 base models and their derived NOFE are given in Table 3 (for completeness we also include results for the base model AlexNet-C100, previously considered). In every case, our NOFE achieves higher accuracy than the corresponding base model. In the case of AlexNet-Quick-C100, the *relative* improvement is a remarkable 20.1%. NIN-C100 was used in [38] as base model to train the hierarchical HD-CNN. The best HD-CNN network from [38] gives 67.38%, whereas we achieve 67.96%. Furthermore, our NOFE is twice as fast at inference (0.0071 vs 0.0147 secs) and it has about half the number of parameters (4.7M vs 9.2M). Finally, according to the online listing of top results on CIFAR100 [1] at the time of this submission, the accuracy of 76.24% obtained by our NOFE built from ResNet56-C100 is the best result ever achieved on this benchmark (the best published accuracy is 72.60% [35] and the best result considering also unpublished work is 75.72% [6]).

Depth and # parameters vs. specialization. Our NOFE differs from the base model in total depth (i.e., number of nonlinearities to compute the output) as well as in number of parameters, because of the additional layers (or residual blocks) in the branches. Here we demonstrate that the improvement does not come from the increased depth or the different number of parameters, but rather from the novel architecture and the procedure to train the experts. To show this, we modify the base networks to match the number of parameters of our NOFE

Table 4. We add layers to the base models in order to match the number of parameters of our NOFE models (the last column reports results with base networks having both the same depth and the same number of parameters as our models). The table reports accuracy (%) on CIFAR100. This study suggests that the accuracy gain of our NOFE does not derive from an increase in depth or number of parameters, but rather from the specialization performed by branches trained on different subsets of classes.

Architecture	Original base model	NOFE	Modified base model matching NOFE # params	Modified base model matching NOFE # params AND depth
AlexNet-C100	54.04	56.24	30.75	50.21
VGG11-C100	68.48	69.27	68.68	68.21
ResNet56-C100	73.52	76.24	73.50	73.88

models. We consider two ways of modifying the base models. In the first case, we add to the original base network $K = 10$ times the number of convolutional layers (or residual blocks) contained in one branch of our NOFE (since we have $K = 10$ branches, each with its own parameters). This produces base networks matching the number of parameters of our NOFE models but being deeper. The other solution is to add to the base model a number of layers (or residual blocks) equal to the number of such layers in one branch of the NOFE, but to increase the number of convolutional filters in each layer to match the number of parameters. This yields modified base networks matching both the total depth and the number of parameters of our models. Table 4 reports the results for 3 distinct base models. The results show unequivocally that our NOFE models outperform base networks of equal learning capacity, even those having same depth.

Finetuning NoFE from generalist vs. learning from scratch Here we want to show that in addition to defining good specialties, the learning of the generalist provides a beneficial pretraining of the trunk of our NOFE. To demonstrate this, we trained NOFE models from scratch (i.e., random weights) using the specialties learned by the generalist. This training setup yields an accuracy of 49.53% when using AlexNet-C100 as base model and 73.95% when using ResNet56-C100. Note that learning the NOFE models from the pretrained generalist yields much better results: 56.2% for AlexNet-C100 and 76.24% for ResNet56-C100. This suggests that the pretraining of the trunk with the generalist is crucial.

4.2 Categorization on ImageNet

In this subsection we evaluate our approach on the ImageNet 2012 classification dataset [7], which includes images of 1000 object categories. The training set consists of 1.28M photos, while the validation set contains 50K images. We train on the training set and use the validation set to assess performance.

Our base model here is the Caffe [20] implementation of AlexNet [23]. It contains 8 learned layers, 5 convolutional and 3 fully connected. As usual, we first train the generalist by simply changing the number of output units in the

last FC layer to K , using our joint learning over network weights and specialty labels. We experimented with two variants: one generalist using $K = 10$ experts, and one with $K = 40$. Both were trained using **fully-balanced** for specialty balancing. The complete NOFE is obtained from the generalist by removing the 3 FC layers and by connecting the last CONV layer to K branches, each with architecture: [CONV:1×256×3],[FC:1024],[FC:1024],[FC:100]. Note that while the base model (and the generalist) use layers of dimensionality 4096 for the first two FC layers, the expert branches use 1024 units in these layers to account for the fact that we have $K = 10$ parallel branches.

We also tested two other ways to define specialties: 1) spectral clustering on the confusion matrix of the base model (on the validation set) and 2) WordNet clustering. The WordNet specialties are obtained by “slicing” the WordNet hierarchy at a depth that intersects $K = 10$ branches of this semantic tree and then aggregating the ImageNet classes of each branch into a different specialty. We also trained a generalist on a random balanced partition of the 1000 classes. Figure 2 shows the classification accuracy of the generalist for these different ways of defining specialties. The accuracy is assessed on the validation set and measures how accurately the generalist recognizes the $K = 10$ specialties as a function of training epochs. We can see that while the CNN trained on the fixed spectral clusters does best in the initial iterations, our generalist (with specialty labels updated every fifth of an epoch) eventually catches up and matches the performance of the spectral generalist. The generalist trained on WordNet clusters and the one trained on random specialties do much worse.

We then built NOFE models from the spectral generalist and our own generalist (we did not train NOFE models from the random or WordNet generalists due to their poor performance). Table 5 shows the accuracy of all these models on the validation set. For each we report top-1 accuracy (averaging over 10 crops per image, taken from the 4 corners and the center, plus mirroring of all of them). We also include results for our approach using $K = 40$ experts. It can be seen that our NOFE with $K = 10$ experts outperforms the base model, yielding a relative improvement of 4.4%. Instead, the NOFE trained on spectral clusters does worse than the base model, despite the good accuracy of the spectral generalist as noted in Fig. 2. We believe that this happens because of the large imbalance of the spectral specialties, which cause certain experts to have classification problems with many more classes than others. Our NOFE with $K = 40$ experts does worse than the one with $K = 10$ experts, possibly because of excessive specialization of the experts or overfitting (see number of parameters in last column). Note that our NOFE with $K = 10$ experts has actually fewer parameters than the base model and yet it outperforms it by a good margin. This indicates that the improvement comes from the structure of our network and the specialization of the experts rather than by larger learning capacity.

In terms of training time, the base model required 7 days on a single NVIDIA K40 GPU. The NOFE with $K = 10$ experts took about 12 days on the same hardware (2 days for the generalist and 10 days for the training of the full model). On CIFAR100 the ratio of the training time between NOFE and base models

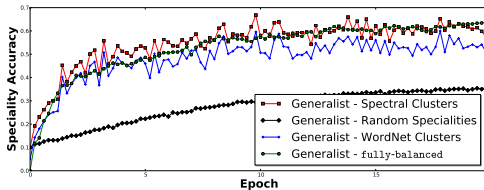


Fig. 2. Generalist accuracy for different definitions of specialties (ours, spectral clusters, random specialties and WordNet clusters). The accuracy is assessed on the validation set for varying training iterations of the generalist.

was about 1.5X (0.5X for the generalist, 1X for the full model). Thus, there is an added computational cost in training our architecture but it is fractional.

Finally, we also evaluated the base model and our NOFE of $K = 10$ experts on the 100K test images of ImageNet, using the test server. The base model achieves a top-1 accuracy of 58.83% while our NOFE yields a recognition rate of 61.48%, thus confirming the improvements seen so far also on this benchmark.

5 Conclusions

In this paper we presented a novel approach that decomposes large multi-class classification into the problem of learning 1) a generalist network distinguishing coarse groupings of classes, called specialties, and 2) a set of expert networks, each devoted to recognize the classes within a specialty. Crucially, our approach learns the specialties and the generalist that recognizes them *jointly*. Furthermore, our approach gives rise to a single tree-structured model that is fine-tuned over the end-objective of recognizing the original set of classes. We demonstrated the generality of our approach by adapting several popular CNNs for image categorization into networks of experts. In each case this translated into an improvement in accuracy at very little added training cost. Software implementing our method and several pretrained NOFE models are available at <http://vlg.cs.dartmouth.edu/projects/nofe/>.

Acknowledgements. We are grateful to Piotr Teterwak for assisting with an early version of this project. We thank Du Tran and Qiang Liu for helpful discussions. This work was funded in part by NSF CAREER award IIS-0952943 and NSF award CNS-1205521. We gratefully acknowledge NVIDIA for the donation of GPUs used for portions of this work.

Table 5. Top-1 accuracy on the ImageNet validation set using AlexNet and our NOFE.

Approach	Top-1 %	# params
Base: AlexNet-Caffe	58.71	60.9M
NOFE, K=10 fully-balanced	61.29	40.4M
NOFE, K=40 fully-balanced	60.85	151.4M
NOFE, K=10 spectral clustering	56.10	40.4M

A Supplementary Material

This supplementary material is organized as follows: in subsection A.1 we discuss a simple Nearest Neighbor (NN) retrieval experiment aimed at illustrating the differences between the features learned by our NOFE and the base model; in subsection A.2 we illustrate some examples of specialties learned from CIFAR100 and ImageNet; in subsection A.3 we demonstrate that finetuning our NOFE from the generalist produces better results than finetuning from a pretrained base model; in subsection A.4 we perform a “depth vs specialization” analysis on the large-scale ImageNet dataset; in subsection A.5 we visualize distributions of specialty sizes obtained by varying the regularization hyperparameter of `elasso`; in subsection A.6 we present a complete specification of all network models presented in the paper together with details about the training procedure; we conclude with subsection A.7 where we discuss our software implementation.

A.1 Nearest Neighbor Retrieval

Figure 3 shows a simple Nearest Neighbor (NN) retrieval experiment on CIFAR100 using features from our experts vs the base model, for the architecture AlexNet-C100. For each query image (first column of the Figure) we perform a NN search in the CIFAR100 test set using as feature representation the last CONV layer of the expert corresponding to the predicted class of the query. The first block of 3 images near the query shows the 3 NNs retrieved in this fashion. The second set of 3 images represents NNs obtained the using as features the last CONV layer of the base model. It can be seen that in many cases the images retrieved with the expert features match the class of the query, while the base model features yield many mismatches.

Figure 4 shows selected retrieval results on the large-scale ImageNet dataset. The first set of 3 images shows NNs obtained using as features the activations from the first FC layer of the expert branch corresponding to the predicted class of the query. The second block of 3 images represents NNs obtained using the first FC layer of the base model. Above each image we report its ImageNet category. We see that the NNs obtained with expert features include many true positives (images of the same class as the query) or, when making mistakes, images of related classes (e.g., a picture of a lynx is retrieved for a cougar query).

A.2 Learned Specialties

In Table 6 we show some of the specialties learned by our generalist on CIFAR100 when using $K = 10$ and AlexNet-C100 as architecture. Most classes within each specialty are semantically or visually similar to each other, but some outliers are also present (e.g., the class “skyscraper” in the first specialty, which otherwise contains natural classes). Table 7 shows that when using $K = 20$ we obtain specialties that are more homogeneous and that make more intuitive sense, since the generalist can perform a finer subdivision of the original $C = 100$ categories. In terms of final categorization accuracy, we have seen that the NOFE with

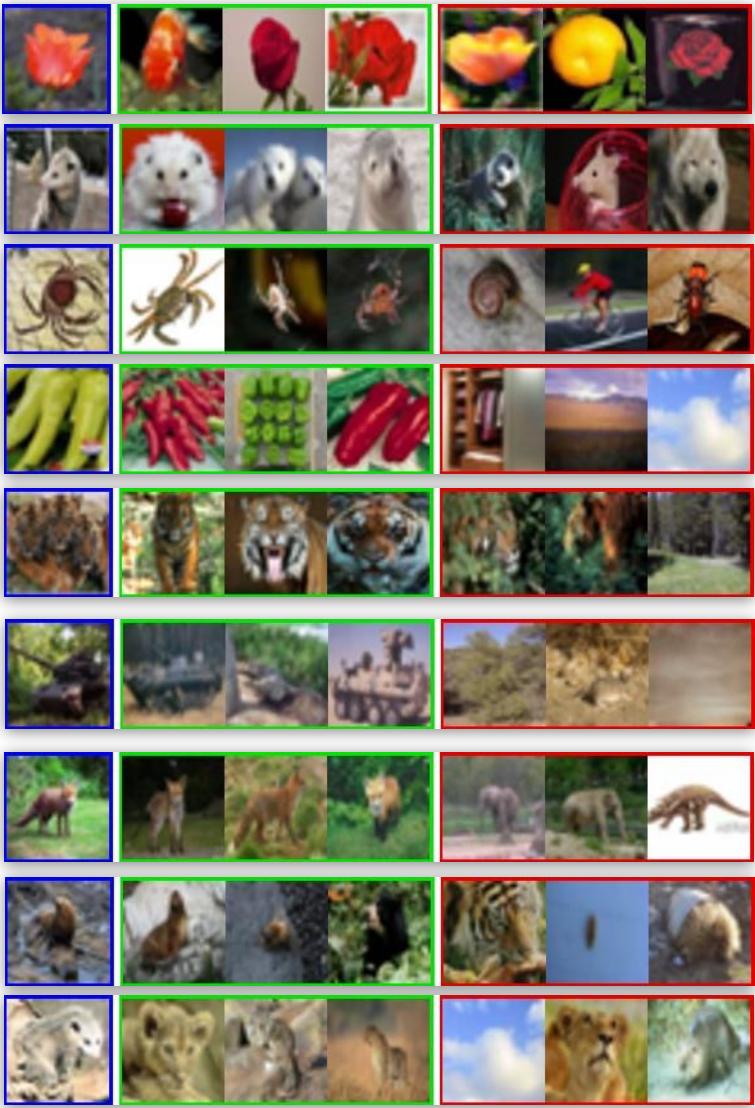


Fig. 3. Nearest Neighbor (NN) retrieval on CIFAR100. The first column shows the query image. The first set of 3 images represents NNs retrieved using our expert features. The second set of 3 images represents NNs obtained with features from the base model.



Fig. 4. Nearest Neighbor (NN) search on ImageNet. The first column shows the query image. The first set of 3 images includes NNs retrieved using as features the activations from the first FC layer of the expert branch associated to the predicted class of the query. The second block of 3 images represents NNs obtained using the first FC layer of the base model. Above each image we report its ImageNet category. Our expert features yield NNs that are semantically related to the query.

Table 6. A few specialties learned by our generalist from CIFAR100 using $K = 10$ with AlexNet-C100 as architecture.

caterpillar, lawn_mower, crab, lizard, forest, maple_tree, oak_tree, pine_tree, willow_tree, skyscraper
pickup_truck, road, streetcar, tank, tractor, train, bus, aquarium_fish, can, house
hamster, leopard, lion, tiger, possum, rabbit, raccoon, squirrel, snail, wolf
bicycle, motorcycle, bottle, camel, cattle, elephant, fox, kangaroo, seal, trout

Table 7. Some example specialties learned from CIFAR100 using $K = 20$ with AlexNet-C100 as architecture. Note how, compared to the case for $K = 10$ shown in Table 6, here the finer subdivision of classes yields specialties that are more homogeneous and that include categories that are more closely related.

maple_tree, oak_tree, pine_tree, willow_tree, palm_tree
apple, cloud, poppy, rose, tulip
dolphin, seal, shark, turtle, whale
baby, boy, girl, man, woman

Table 8. A few specialties learned by our generalist from ImageNet using AlexNet as architecture and $K = 10$.

can opener, tinopener, carpenter’s kit, cassette, cassette player, cellular phone, hand-held computer, iPod, joystick, speaker, computer mouse, parking meter, pay-phone, pay-station, photo copier, polaroid camera, printer, remote control, scale, stove, electric switch, . . .
cock, hen, black grouse, macaw, drake, flamingo, great pyrenees, standard poodle, ice bear, polar bear, Arabian camel, baseball, golfball, lifeboat, pencil sharpener, schoolbus, streetcar, tram, trolley, trolley car, trolley bus, trolley coach, lemon, rapeseed, yellow lady-slipper, rosehip, coralfungus, agaric, . . .
hyena, zebra, Indian elephant, African elephant, lion fish, spiny lobster, seacraw-fish, cray fish, rock lobster, chain, chain mail, ring mail, necklace, modem, packet, puck, hockey puck, comicbook, crossword puzzle, menu, . . .
Bordercollie, GreaterSwissMountaindog, Bernesemountaindog, Appenzeller, Entle-Bucher, affenpinscher, monkeypinscher, monkeydog, West Highland white terrier, white wolf, Arctic wolf, Arctic fox, white fox, Americanblackbear, sorrel, ox, water buffalo, water ox, Asiatic buffalo, bison, bearskin, busby, shako, guillotine, harp, horse-cart, hourglass, megalith, oxcart, snowplow, thatch, . . .

$K = 20$ does not provide better results compared to $K = 10$. This is most likely due to the higher number of parameters in the configuration with $K = 20$, which may cause overfitting.

In Table 8, we show some of the specialists learned by our generalist on ImageNet, using $K = 10$ and AlexNet as architecture.

A.3 Finetuning from the base model

Our NOFE performs learning from scratch and does not require the learning of the base model in order to train the generalist and the experts. This is advantageous as it leads to a faster and more streamlined training procedure. However,

one may wonder if learning the NOFE from the pretrained base model may actually lead to better performance. We attempted this experiment by finetuning the generalist from the base model (AlexNet-C100). The resulting generalist was then used to train the full NOFE, as usual. The accuracy achieved with this setup is 55.5%, thus inferior to the 56.2% produced when learning from scratch. This suggests that the features learned by the base model by solving the hard classification over C classes are providing a poor initialization for the generalist and the subsequent expert branches, which is instead the approach used in prior expert-based networks [17,37,38].

A.4 Depth vs Specialization

Our NOFE typically has one convolutional layer more than the base model, as we include a convolutional layer in each branch. In the experimental section of our paper we discussed the results of a CIFAR100 experiment, which shows that the improvement in accuracy achieved by our approach is not due to the increased depth of the model but rather from its structure. Here we report the same experiment but for the case of ImageNet: we trained a variant of the AlexNet-Caffe base model that has one additional convolutional layer (CONV:1×256×3) such that this network has total depth exactly equal to the depth of the NOFE presented in Table 5 of our paper. We found that this deeper variant of the base model yields an accuracy of 56.91%, thus lower than the shallower base network and much lower than the accuracy of 61.29% achieved by our NOFE. This indicates once again that the critical improvement in our approach comes from the specialization performed by the experts rather than from the additional layer.

A.5 Distribution of specialty sizes

In the paper we demonstrated the importance of balancing the size of the specialties. One of the methods investigated is **elasso**, which encourages softly the constraint over the size of specialties. The behavior of **elasso** is defined by hyperparameter λ which trades off the importance between having balanced specialties and good categorization accuracy over them. Figure 5 shows the different distributions in specialty size obtained for different values of λ with $K = 10$ experts. Small values of λ (e.g., 100 or 200) produce specialties that are highly uneven in size. Conversely, a large value of λ yields perfectly balanced classes.

A.6 Network Specifications

In this subsection, we provide full specification of all networks used in our experiments on the CIFAR100 and ImageNet datasets. We also discuss the details of the learning policy for each model.

Tables 9, 10, 11, 12, and 13 list the specifications of the 4 architectures used on CIFAR100: AlexNet-C100, AlexNet-Quick-C100, VGG11-C100, NIN-C100, and

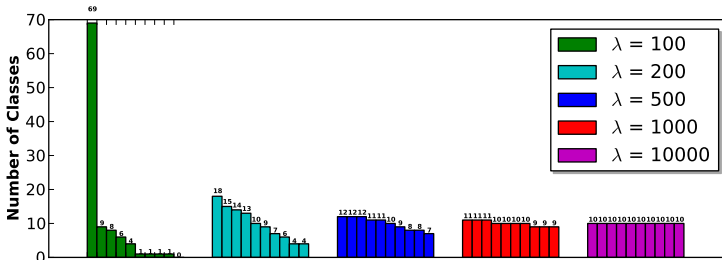


Fig. 5. Distribution of specialty sizes learned by **elasso** for different λ values (100, 200, 500, 1000, and 10000) using $K = 10$ experts. For each λ the specialties are sorted in decreasing size. Small values of λ yield highly imbalanced specialties, while large values of λ force specialties to have nearly equal size.

ResNet56-C100. Table 14 provides the details of the AlexNet-Caffe architecture used for the large-scale ImageNet experiment (Table 4 in the paper).

Each table lists the specification of a network in two columns. The first column provides the details of both the generalist and the base model, which always share the same architecture, except for the number of units in the last fully connected layer and the training policy. The second column shows the Network of Experts (NOFE) obtained by removing the last fully connected layer of the generalist and by attaching to it K expert branches. The architecture of the branches is listed in the yellow blocks.

Each entry in the tables represents a stack of layers, i.e., a set of one or more layers (convolutional or fully connected) connected directly to each other, and usually followed by a pooling layer. In all architectures we used rectified linear units (ReLU). In the following, we illustrate the notation used to describe each layer type.

Notation:

- Convolutional layer:

CONV: $\langle \text{number of layers} \rangle \times \langle \text{number of filters} \rangle \times \langle \text{filter size} \rangle$

Example: CONV: $1 \times 32 \times 5$ means 1 convolutional layer with 32 filters of size 5×5 .

- Pooling layer:

POOL: $\langle \text{filter size} \rangle, \langle \text{stride size} \rangle, \langle \text{pooling type} \rangle$

Example: POOL: 3,2,Max: maximum pooling layer of filter size 3 and stride 2. Two types are used: Max (Maximum), Ave (Average).

- LRN: Local Response Normalization layer

- Fully Connected layer:

FC: $\langle \text{size} \rangle$ Example: FC: C indicates a fully connected layer with C output

units. We use C to denote the number of classes and K to indicate the number of specialties. For simplicity, we assume that the architecture of the NOFE illustrated in all tables is based on the **fully-balanced** method to train the generalist. This implies that each speciality contains the same number of classes (C/K).

– Residual block:

Residual blocks are used in Residual Networks [14]. We use big braces to indicate residual blocks. For example, the following notation denotes a concatenation of 9 residual blocks, each of which consists of a CONV:1×64×3 layer and another CONV:1×64×3 layer, with batch normalization and scaling layers in between, as proposed in [14]:

$$\left\{ \begin{array}{l} \text{CONV: } 1 \times 64 \times 3 \\ \text{CONV: } 1 \times 64 \times 3 \end{array} \right\} \times 9$$

– Expert branch:

An expert branch is a stack of one or more layers. A NOFE includes K parallel branches connected to the trunk. We highlight the expert branches with yellow color in each table, and denote the i -th expert branch with “Expert^(i) → ”.

Table 9. AlexNet-C100 (trained on CIFAR100)

Generalist and Base Model	Network of Experts (NofE)
CONV: $1 \times 32 \times 5$ POOL: 3,2,Max LRN	CONV: $1 \times 32 \times 5$ POOL: 3,2,Max LRN
CONV: $1 \times 32 \times 5$ POOL: 3,2,Ave LRN	CONV: $1 \times 32 \times 5$ POOL: 3,2,Ave LRN
CONV: $1 \times 64 \times 5$ POOL: 3,2,Ave	CONV: $1 \times 64 \times 5$ POOL: 3,2,Ave LRN
	Expert ⁽ⁱ⁾ \rightarrow CONV: $1 \times 64 \times 5$ Expert ⁽ⁱ⁾ \rightarrow POOL: 3,2,Ave
FC: K (Generalist) FC: C (Base Model)	Expert ⁽ⁱ⁾ \rightarrow FC: C/K
Data preprocessing and augmentation	
Input: 32×32 Image mean subtraction	Input: 32×32 Image mean subtraction
Learning Policy	
<i>Generalist:</i> Learning rate: 0.001 (Fixed) 0.001 : 60 Epochs Momentum: 0.9 Weight decay: 0.004 Weight initialization: Random MiniBatch size: 100 <i>Base Model:</i> Learning rate: 0.001 (lowered twice) 140 Epochs Momentum: 0.9 Weight decay: 0.004 Weight initialization: Random MiniBatch size: 100	Learning rate: 0.001 till 0.00001 (lowered twice) 0.001 : 120 Epochs 0.0001 : 10 Epochs 0.00001 : 10 Epochs Momentum: 0.9 Weight decay: 0.004 Weight initialization: Random MiniBatch size: 100

Table 10. AlexNet-Quick-C100 (trained on CIFAR100)

Generalist and Base Model	Network of Experts (NoE)
CONV: $1 \times 32 \times 5$ POOL: 3,2,Max	CONV: $1 \times 32 \times 5$ POOL: 3,2,Max
CONV: $1 \times 32 \times 5$ POOL: 3,2,Ave	CONV: $1 \times 32 \times 5$ POOL: 3,2,Ave
CONV: $1 \times 64 \times 5$ POOL: 3,2,Ave	CONV: $1 \times 64 \times 5$ POOL: 3,2,Ave
	Expert ⁽ⁱ⁾ \rightarrow CONV: $1 \times 64 \times 5$ Expert ⁽ⁱ⁾ \rightarrow POOL: 3,2,Ave
FC: 64	Expert ⁽ⁱ⁾ \rightarrow FC: 64
FC: K (Generalist) FC: C (Base Model)	Expert ⁽ⁱ⁾ \rightarrow FC: C/K
Data preprocessing and augmentation	
Input: 32×32 Image mean subtraction	Input: 32×32 Image mean subtraction
Learning Policy	
<i>Generalist:</i> Learning rate: 0.001 (Fixed) 0.001 : 8 Epochs Momentum: 0.9 Weight decay: 0.004 Weight initialization: Random MiniBatch size: 100 <i>Base Model:</i> Learning rate: 0.001 (lowered twice) 12 Epochs Momentum: 0.9 Weight decay: 0.004 Weight initialization: Random MiniBatch size: 100	Learning rate: 0.001 till 0.00001 (lowered twice) 0.001 : 8 Epochs 0.0001 : 2 Epochs 0.00001 : 2 Epochs Momentum: 0.9 Weight decay: 0.004 Weight initialization: Random MiniBatch size: 100

Table 11. VGG11-C100 (trained on CIFAR100)

Generalist and Base Model	Network of Experts (NofE)
CONV: $2 \times 64 \times 3$ POOL: 2,2,Max	CONV: $2 \times 64 \times 3$ POOL: 2,2,Max
CONV: $2 \times 128 \times 3$ POOL: 2,2,Max	CONV: $2 \times 128 \times 3$ POOL: 2,2,Max
CONV: $4 \times 256 \times 3$ POOL: 2,2,Max	CONV: $4 \times 256 \times 3$ POOL: 2,2,Max
	Expert ⁽ⁱ⁾ \rightarrow CONV: $2 \times 256 \times 3$ Expert ⁽ⁱ⁾ \rightarrow POOL: 2,2,Max
FC: 1024	Expert ⁽ⁱ⁾ \rightarrow FC: 512
FC: 1024	Expert ⁽ⁱ⁾ \rightarrow FC: 512
FC: K (Generalist) FC: C (Base Model)	Expert ⁽ⁱ⁾ \rightarrow FC: C/K
Data preprocessing and augmentation	
4 zeros padding on each side, then 32×32 crops Image mean subtraction Image mirroring	4 zeros padding on each side, then 32×32 crops Image mean subtraction Image mirroring
Learning Policy	
<i>Generalist:</i> Learning rate: 0.001 (Fixed) 0.001 : 60 Epochs Momentum: 0.9 Weight decay: 0.0005 Weight initialization: Xavier [11] MiniBatch size: 128 <i>Base Model:</i> Learning rate: 0.001 (lowered twice) 200 Epochs Momentum: 0.9 Weight decay: 0.0005 Weight initialization: Xavier [11] MiniBatch size: 128	Learning rate: 0.001 till 0.00001 (lowered twice) 0.001 : 120 Epochs 0.0001 : 10 Epochs 0.00001 : 10 Epochs Momentum: 0.9 Weight decay: 0.0005 Weight initialization: Xavier [11] MiniBatch size: 128

Table 12. NIN-C100 (trained on CIFAR100)

Generalist and Base Model	Network of Experts (NofE)
CONV: $1 \times 192 \times 5$ CONV: $1 \times 160 \times 1$ CONV: $1 \times 96 \times 1$ POOL: 3,2,Max	CONV: $1 \times 192 \times 5$ CONV: $1 \times 160 \times 1$ CONV: $1 \times 96 \times 1$ POOL: 3,2,Max
CONV: $1 \times 192 \times 5$ CONV: $1 \times 192 \times 1$ CONV: $1 \times 192 \times 1$ POOL: 3,2,Max	CONV: $1 \times 192 \times 5$ CONV: $1 \times 192 \times 1$ CONV: $1 \times 192 \times 1$ POOL: 3,2,Max
CONV: $1 \times 192 \times 3$ CONV: $1 \times 192 \times 1$	CONV: $1 \times 192 \times 3$ CONV: $1 \times 192 \times 1$
	Expert ⁽ⁱ⁾ \rightarrow CONV: $1 \times 192 \times 3$ Expert ⁽ⁱ⁾ \rightarrow CONV: $1 \times 192 \times 1$
CONV: $1 \times K \times 1$ (Generalist) CONV: $1 \times C \times 1$ (BaseModel)	Expert ⁽ⁱ⁾ \rightarrow CONV: $1 \times (C/K) \times 1$
POOL: 8,1,AVE (Generalist) POOL: 8,1,AVE (Base-Model)	Expert ⁽ⁱ⁾ \rightarrow POOL: 6,1,AVE
Data preprocessing and augmentation	
Input crop: 26×26 Image mean subtraction Image mirroring	Input crop: 26×26 Image mean subtraction Image mirroring
Learning Policy	
<i>Generalist:</i> Learning rate: 0.01 (Fixed) 0.01 : 200 Epochs Momentum: 0.9 Weight decay: 0.001 Weight initialization: Random MiniBatch size: 100 <i>Base Model:</i> Learning rate: 0.01 (lowered twice) 0.01 : 220 Epochs 0.001 : 10 Epochs 0.0001 : 30 Epochs Momentum: 0.9 Weight decay: 0.001 Weight initialization: Random MiniBatch size: 100	Learning rate: 0.01 till 0.0001 (lowered twice) 0.01 : 98 Epochs 0.001 : 120 Epochs 0.0001 : 10 Epochs Momentum: 0.9 Weight decay: 0.001 Weight initialization: Random MiniBatch size: 100

Table 13. ResNet56-C100 (trained on CIFAR100)

Generalist and Base Model	Network of Experts (NofE)
CONV: $1 \times 64 \times 3$	CONV: $1 \times 64 \times 3$
$\left\{ \begin{array}{l} \text{CONV: } 1 \times 64 \times 3 \\ \text{CONV: } 1 \times 64 \times 3 \end{array} \right\} \times 9$	$\left\{ \begin{array}{l} \text{CONV: } 1 \times 64 \times 3 \\ \text{CONV: } 1 \times 64 \times 3 \end{array} \right\} \times 9$
$\left\{ \begin{array}{l} \text{CONV: } 1 \times 128 \times 3 \\ \text{CONV: } 1 \times 128 \times 3 \end{array} \right\} \times 9$	$\left\{ \begin{array}{l} \text{CONV: } 1 \times 128 \times 3 \\ \text{CONV: } 1 \times 128 \times 3 \end{array} \right\} \times 9$
$\left\{ \begin{array}{l} \text{CONV: } 1 \times 256 \times 3 \\ \text{CONV: } 1 \times 256 \times 3 \end{array} \right\} \times 9$	$\left\{ \begin{array}{l} \text{CONV: } 1 \times 256 \times 3 \\ \text{CONV: } 1 \times 256 \times 3 \end{array} \right\} \times 9$
POOL: 7,1,Ave	$\text{Expert}^{(i)} \rightarrow \left\{ \begin{array}{l} \text{CONV: } 1 \times 256 \times 3 \\ \text{CONV: } 1 \times 256 \times 3 \end{array} \right\} \times 1$ $\text{Expert}^{(i)} \rightarrow \text{POOL: 7,1,Ave}$
FC: K (Generalist) FC: C (Base Model)	$\text{Expert}^{(i)} \rightarrow \text{FC: } C/K$
Data preprocessing and augmentation	
Crop size: 28 Image mean subtraction Image mirroring	Crop size: 28 Image mean subtraction Image mirroring
Learning Policy	
<i>Generalist:</i> Learning rate: 0.01 (Fixed) 0.01 : 12 Epochs Momentum: 0.9 Weight decay: 0.0001 Weight initialization: MSRA [15] MiniBatch size: 128 <i>Base Model:</i> Learning rate: 0.1 (lowered twice) 60 Epochs Momentum: 0.9 Weight decay: 0.0001 Weight initialization: MSRA [15] MiniBatch size: 128	Learning rate: 0.1 till 0.001 (lowered twice) 0.1 : 30 Epochs 0.01 : 15 Epochs 0.001 : 15 Epochs Momentum: 0.9 Weight decay: 0.0005 Weight initialization: MSRA [15] MiniBatch size: 128

Table 14. AlexNet-Caffe (trained on ImageNet)

Generalist	Network of Experts (NoE)
CONV: $2 \times 96 \times 11$ LRN POOL: 3,2,Max	CONV: $2 \times 96 \times 11$ LRN POOL: 3,2,Max
CONV: $2 \times 384 \times 3$ CONV: $1 \times 256 \times 3$ LRN POOL: 3,2,Max	CONV: $2 \times 384 \times 3$ CONV: $1 \times 256 \times 3$ LRN POOL: 3,2,Max
	Expert ⁽ⁱ⁾ \rightarrow CONV: $1 \times 256 \times 3$ Expert ⁽ⁱ⁾ \rightarrow POOL: 3,2,Max
FC: 4096	Expert ⁽ⁱ⁾ \rightarrow FC: 1024
FC: 4096	Expert ⁽ⁱ⁾ \rightarrow FC: 1024
FC: K (Generalist) FC: C (Base Model)	Expert ⁽ⁱ⁾ \rightarrow FC: C/K
Data preprocessing and augmentation	
Crop size: 227 Image mean subtraction Image mirroring	Crop size: 227 Image mean subtraction Image mirroring
Learning Policy	
<i>Generalist:</i> Learning rate: 0.01 (Fixed) 0.01: 20 Epochs Momentum: 0.9 Weight decay: 0.0005 Weight initialization: Random MiniBatch size: 256 <i>Base Model:</i> Learning rate: 0.01 (lowered twice) 80 Epochs Momentum: 0.9 Weight decay: 0.0005 Weight initialization: Random MiniBatch size: 256	Learning rate: 0.01 till 0.0001 (lowered twice) 0.001 : 20 Epochs 0.001 : 20 Epochs 0.0001 : 20 Epochs Momentum: 0.9 Weight decay: 0.0005 Weight initialization: Random MiniBatch size: 256

A.7 Software Implementation

Our software implementation is based on the Caffe library [20]. In order to implement our Network of Experts, we have made changes to the Caffe library, including new development of layers, solvers, and tools. Software implementing our method and several pretrained NOFE models are available at <http://vlg.cs.dartmouth.edu/projects/nofe/>.

References

1. Benenson, R.: Classification datasets results, http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html
2. Bengio, S., Weston, J., Grangier, D.: Label embedding trees for large multi-class tasks. In: Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada. pp. 163–171 (2010)
3. Bergamo, A., Torresani, L.: Meta-class features for large-scale object categorization on a budget. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012. pp. 3085–3092 (2012)
4. Biederman, I.: Recognition-by-components: a theory of human understanding. *Psychological Review* 94:115-147 (1987)
5. Chang, X., Nie, F., Ma, Z., Yang, Y.: Balanced k-means and min-cut clustering. CoRR abs/1411.6235 (2014), <http://arxiv.org/abs/1411.6235>
6. Clevert, D., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). CoRR abs/1511.07289 (2015), <http://arxiv.org/abs/1511.07289>
7. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Li, F.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA. pp. 248–255 (2009)
8. Deng, J., Krause, J., Berg, A.C., Li, F.: Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012. pp. 3450–3457 (2012)
9. Erhan, D., Bengio, Y., Courville, A.C., Manzagol, P., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research* 11, 625–660 (2010)
10. Gao, T., Koller, D.: Discriminative learning of relaxed hierarchy for large-scale visual recognition. In: ICCV (2011)
11. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011. pp. 315–323 (2011)
12. Griffin, G., Perona, P.: Learning and using taxonomies for fast visual categorization. In: 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA (2008)

13. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III. pp. 346–361 (2014)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR abs/1512.03385 (2015)
15. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015. pp. 1026–1034 (2015)
16. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Improving neural networks by preventing co-adaptation of feature detectors. CoRR abs/1207.0580 (2012)
17. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. CoRR abs/1503.02531 (2015)
18. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. pp. 448–456 (2015)
19. Jia, Y., Abbott, J.T., Austerweil, J.L., Griffiths, T.L., Darrell, T.: Visual concept learning: Combining machine vision and bayesian generalization on concept hierarchies. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. pp. 1842–1850 (2013)
20. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
21. Kotschieder, P., Fiterau, M., Criminisi, A., Rota Bulò, S.: Deep neural decision forests. In: The IEEE International Conference on Computer Vision (ICCV) (December 2015)
22. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009), technical Report <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>
23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. pp. 1106–1114 (2012)
24. Lee, C., Xie, S., Gallagher, P.W., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015 (2015)
25. Lee, H., Grosse, R.B., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009. pp. 609–616 (2009)
26. Lin, Min, Chen, Q., Yan, S.: Network in network. In: International Conference on Learning Representations, 2014 (arXiv:1409.1556). (2014)
27. Liu, B., Sadeghi, F., Tappen, M.F., Shamir, O., Liu, C.: Probabilistic label trees for efficient large scale image classification. In: 2013 IEEE Conference on Computer

- Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013. pp. 843–850 (2013)
28. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. *Proc. ICML* 30, 1 (2013)
 29. Marszalek, M., Schmid, C.: Constructing category hierarchies for visual recognition. In: *Computer Vision - ECCV 2008*, 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part IV. pp. 479–491 (2008)
 30. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: *Advances in Neural Information Processing Systems 14* [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]. pp. 849–856 (2001)
 31. Ordonez, V., Deng, J., Choi, Y., Berg, A.C., Berg, T.L.: From large scale image categorization to entry-level categories. In: *IEEE International Conference on Computer Vision, ICCV 2013*, Sydney, Australia, December 1-8, 2013. pp. 2768–2775 (2013)
 32. Salakhutdinov, R., Torralba, A., Tenenbaum, J.B.: Learning to share visual appearance for multiclass object detection. In: *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011*, Colorado Springs, CO, USA, 20-25 June 2011. pp. 1481–1488 (2011)
 33. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556 (2014)
 34. Sivic, J., Russell, B.C., Zisserman, A., Freeman, W.T., Efros, A.A.: Unsupervised discovery of visual object class hierarchies. In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, 24-26 June 2008, Anchorage, Alaska, USA (2008)
 35. Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M.M.A., Prabhat, Adams, R.P.: Scalable bayesian optimization using deep neural networks. In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, Lille, France, 6-11 July 2015. pp. 2171–2180 (2015)
 36. Srivastava, N., Salakhutdinov, R.: Discriminative transfer learning with tree-based priors. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. pp. 2094–2102 (2013)
 37. Warde-Farley, D., Rabinovich, A., Anguelov, D.: Self-informed neural network structure learning. *CoRR* abs/1412.6563 (2014)
 38. Yan, Z., Zhang, H., Piramuthu, R., Jagadeesh, V., DeCoste, D., Di, W., Yu, Y.: HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. In: *2015 IEEE International Conference on Computer Vision, ICCV 2015*, Santiago, Chile, December 7-13, 2015. pp. 2740–2748 (2015)
 39. Zhou, Y., Jin, R., Hoi, S.C.H.: Exclusive lasso for multi-task feature selection. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010*, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010. pp. 988–995 (2010)